On Tevatron Tune Fitter/Tracker, Status Reports and Plans

Paul Lebrun

Fermilab

May 9 2003

The team…

- *Jim Patrick, Charlie Briegel, Ron Rechenmaker* for Control and D.A. software
- *Dean Still, Charlie Briegel* HP3561a installation & support, access to VSA tune data.
- *John Marraffino*, offline software and ROOT interfacing
- *Vladimir Shiltsev, & TeV dept*, for their support and patience in MCR..

Outline

- Goal and Scope of this project.
- Brief description Algorithm used in fitting, and C++/Java implementation.
- Examples of fits
- Can this method be used at ~ 20 Hz instead of a fraction of 1Hz ?
    - Yes, based solely on ad-hoc convolution and simple extremum detection.
    - Test algorithm written, performance O.K., need data !

- *Note :By not too distant training, I am a High Energy Physics, Accelerator Physics is something new exciting. → expect some naïve ideas/statements!.*

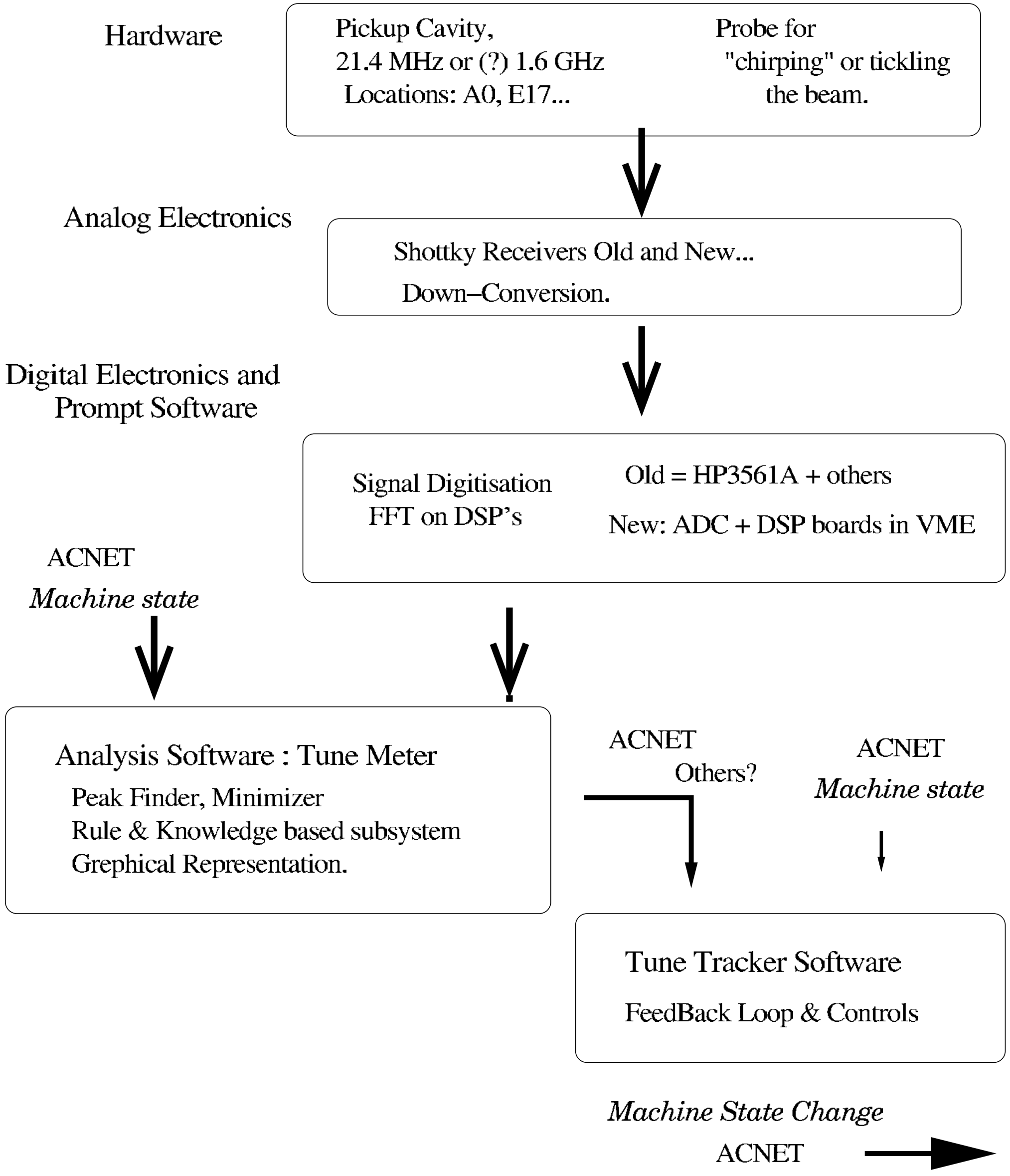
Tevatron Tune Tracking: Goal & Scope Written January 30 2003.. *Edited this week.*

- Automatic fits of the Tune Spectrum Analyzer data seems a difficult task, as it is just a mess of broad bump, narrow signals, and "mostly noise" (especially for coalesced beams)
- Goal of a Tune Meter : express "the art of picking the right line" into a reproducible algorithm that can be implemented on a modern computer, and can be run at ~ 1 Hz.
  - To improve the overall reliability of such measurements. ***Done***
  - Reduce clock time to doing such measurements ***Not demonstrated, real chance it will happen***
  - Allow the implementation a tune tracker, based a straight feedback loop using this tune meter. **To be considered**

Tevatron Tune Tracking: Goal & Scope Written January 30 2003.. *Edited this week., II*

- Scope:
  - Short term: Using existing equipment, (21.4 MHz Shottky, HP3561a) and new software (C++, Java, Root,..) ***Done, v1.0***
  - Long Term: dedicated Front-end subsystem with better digitization and FFT on DSP, refine analysis software… ***Under Construction….***

Tune Meter/Tracker : A simplified "System" View



Hardware

Pickup Cavity,
21.4 MHz or (?) 1.6 GHz
Locations: A0, E17...

Probe for
"chirping" or tickling
the beam.

Analog Electronics

Shottky Receivers Old and New...
Down–Conversion.

Digital Electronics and
Prompt Software

Signal Digitisation
FFT on DSP's

Old = HP3561A + others

New: ADC + DSP boards in VME

ACNET
Machine state

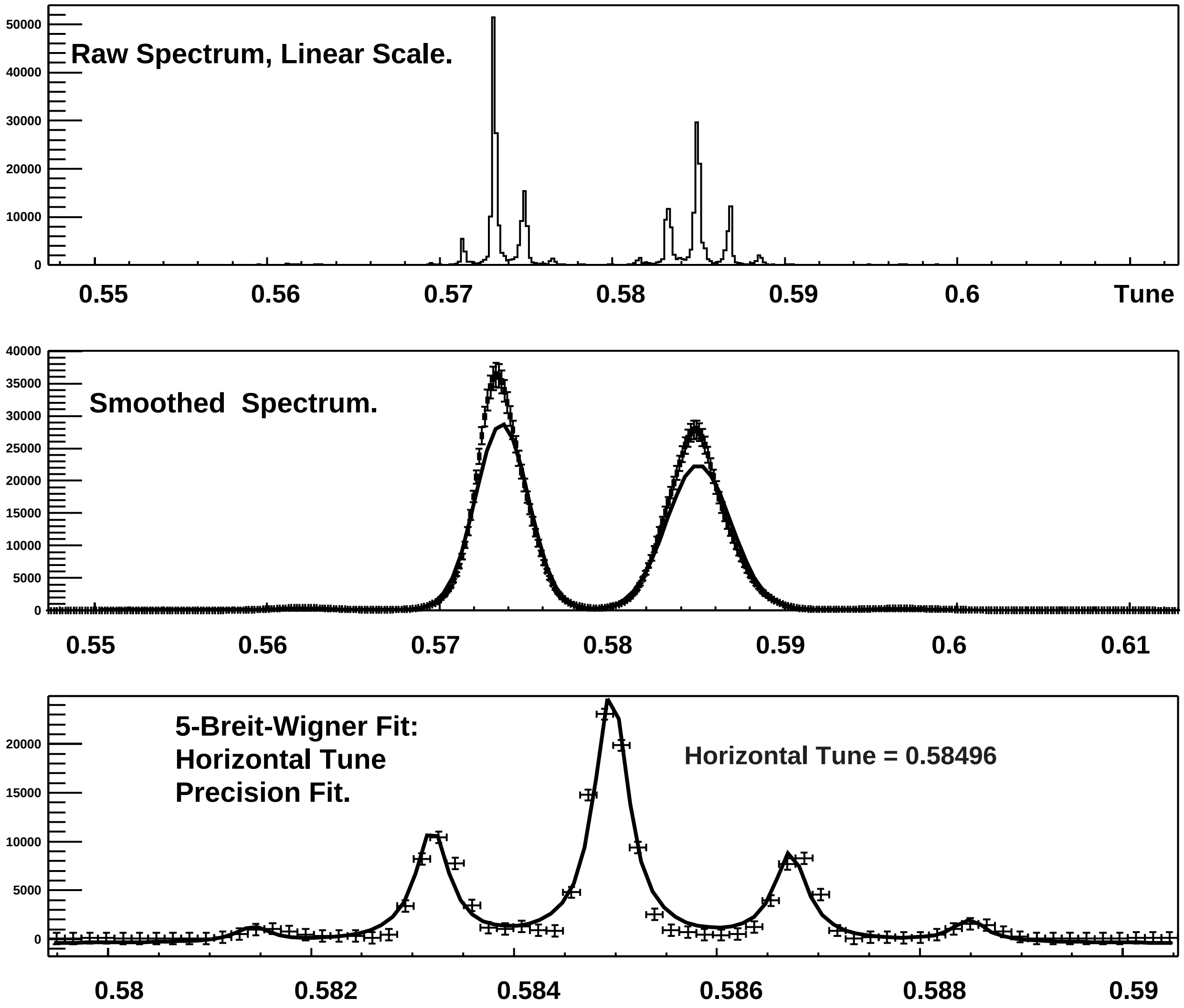
Analysis Software : Tune Meter
Peak Finder, Minimizer
Rule & Knowledge based subsystem
Grephical Representation.

ACNET
Others?

ACNET
Machine state

Tune Tracker Software
FeedBack Loop & Controls

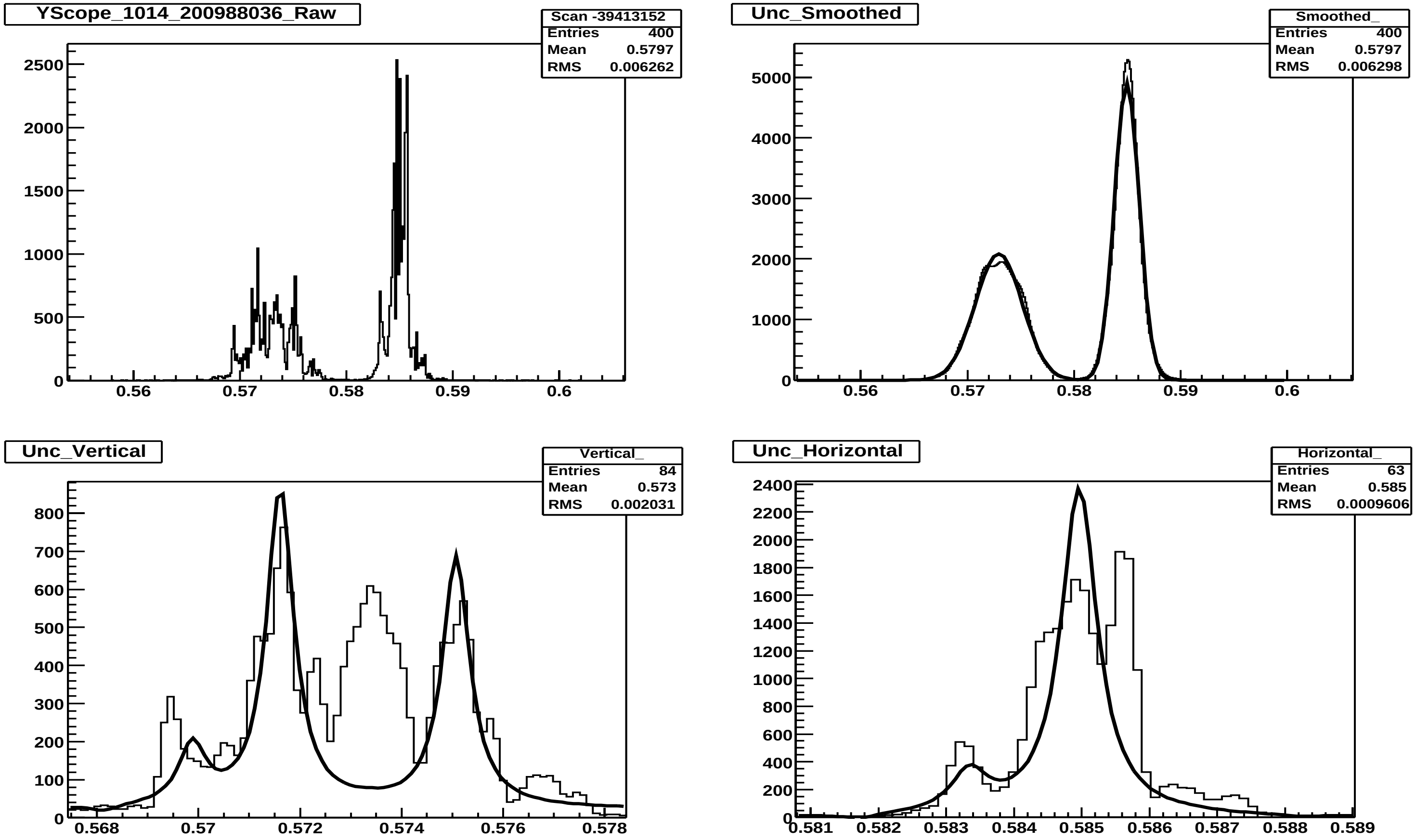
Machine State Change
ACNET


Raw Spectrum, Linear Scale.
50000
40000
30000
20000
10000
0
0.55
0.56
0.57
0.58
0.59
0.6
Tune
Smoothed Spectrum.
40000
35000
30000
25000
20000
15000
10000
5000
0
0.55
0.56
0.57
0.58
0.59
0.6
0.61
5-Breit-Wigner Fit:
Horizontal Tune
Precision Fit.
Horizontal Tune = 0.58496
20000
15000
10000
5000
0
0.58
0.582
0.584
0.586
0.588
0.59

Algorithms..Uncoalesced..

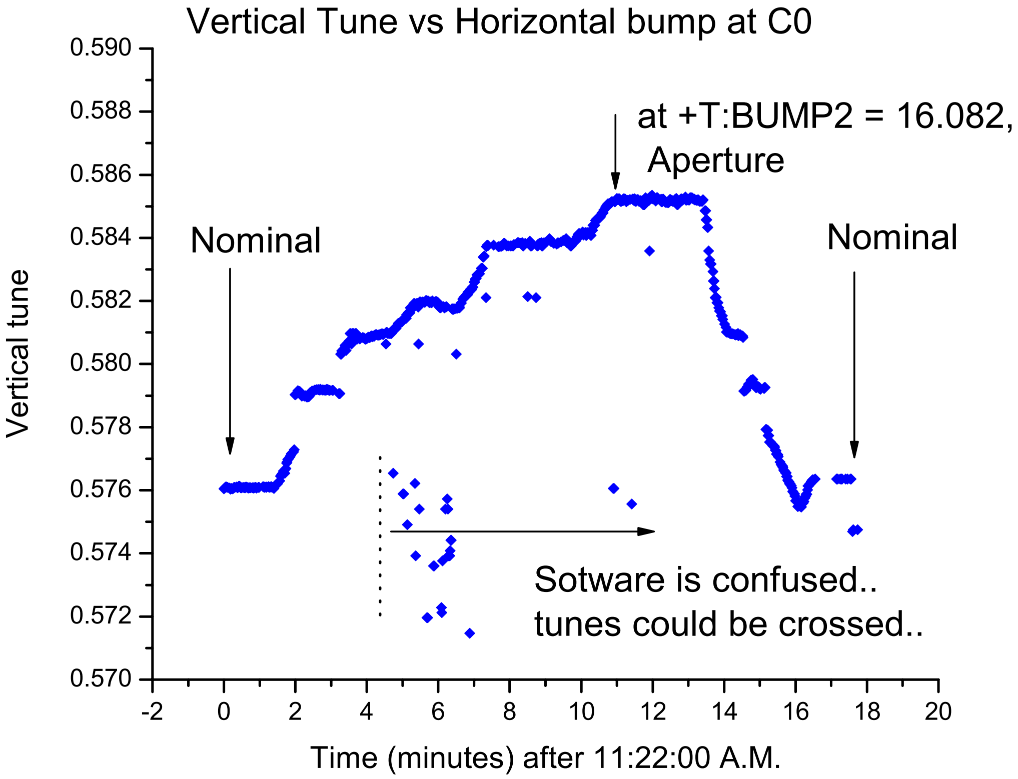
- First, Histogram, on a linear Y scale.
- Scale such the noise level (~-80to 70 db) corresponds to few counts per bin.
- Smear (or smooth), on a big scale: every bin content is spread, Gaussian wise, to neighboring bins. This is just a Gaussian convolution or "transform"
- Fit Two Gaussians. This determines the broad value of the Horizontal and Vertical tunes.
- Make two distinct new histograms, one for each region, using the original data.
- Smooth, Cern algorithm, two times.
- Fit with 5 Breit-Wigners, with same widths and same frequency splitting between satellites and main line.

Spectrum as “Things Change.”23:04, Dec 11



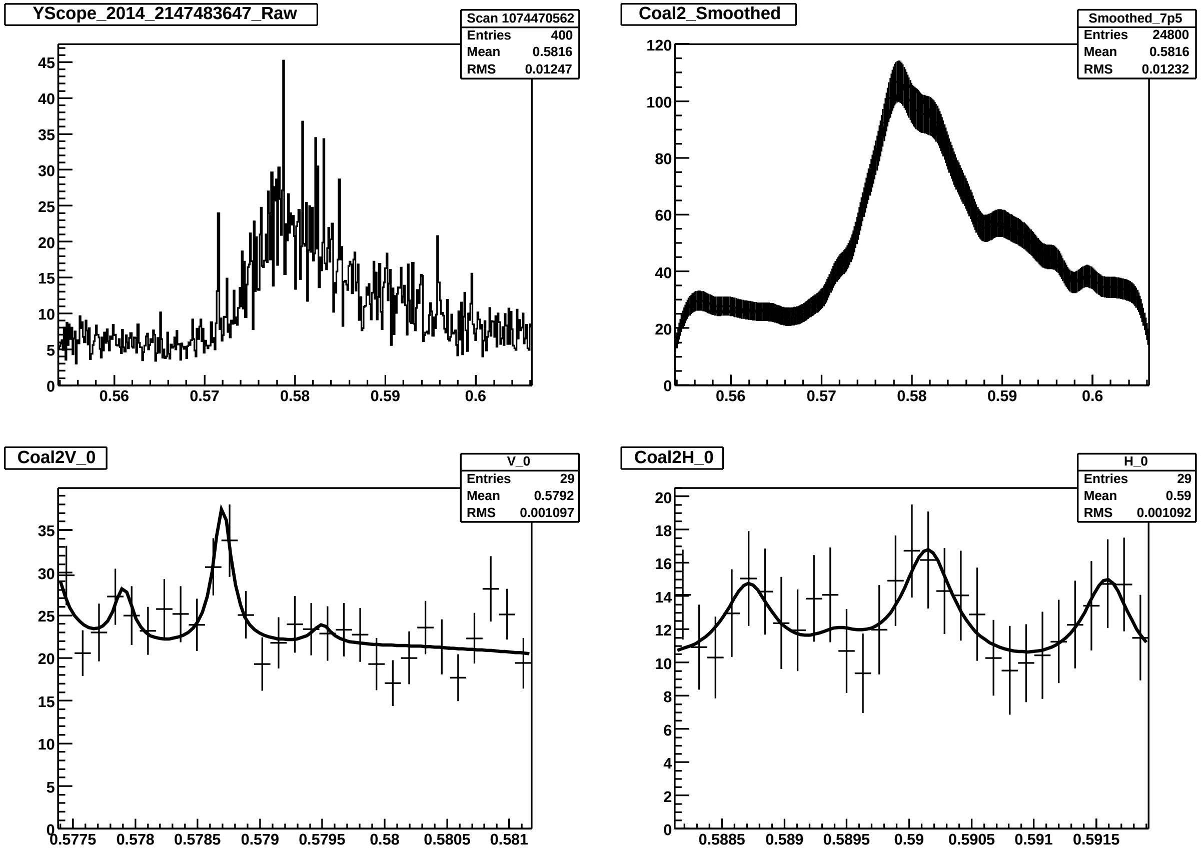
Despite missed bumps, Synch split, H = 0.0017312, V = 0.0016207, Predicted = 0.00166

Vertical Tunes vs Bump position at C0 (Parasitic)



- **Very sensitive to horizontal position.**
- **Caveat (again) : tunes did cross wile doing the scan and the software is confused.**

Coalesced, p-bar beams is much harder!



- Data taken on Dec. 16 2002, 11:38 A.M. (store 2078, ~ 2 hours into the store).
- Nothing but noise lines at this point???
- There is more than one tune !
- How do we establish a signal?
- Note : these lines are clearly beam related!

Fast Algorithm: From a fraction of 1Kz to ~20 Hz.

- 1 Hz not quite good enough with respect to changes occurring during the ramp, if this fitter ought in a feed-back loop.
- Can a passive system, with a fast tune fitter work, work for the Tevatron : Yes, it should work. At 20 Hz, or faster..
- The question is: will be it be precise enough? It can't work better than ~ 2 π (20 Hz/27 KHz) ~ 4.5 10^{-3} The convolution process will make it worse by (?) $\sqrt{2}$ at best..
- Which is not quite good enough for the Run II TeV, given our limited dynamic aperture and relatively large betatron coupling (min. tune split of ~ 0.003), and the constraints from lattice (We run at ($\nu_x - n_y$) ~ 0.009)
- ***Yet, we should try to speed-up these fitting algorithms!! It is a good idea…***

A Fast Algorithm: Determine the extremum of a Gaussian Convolution of the signal.

- If guaranteed speed is an issue, “real time” would be nice.
- If “real-time”, fixed – or almost fixed – number of operations!.
- A single convolution with fixed parameters might work..
  - Two embedded loops: for all channels, amplitude is an integral
  - Inner loop limited
  - fixed coefficients in sum in this inner loop.
- Then, once convoluted, don’t fit, simply look for the extremum (a) .
  - Using numerical derivative to locate the extrema That’s the tune location. It works, provided the noise frequency is high enough with respect to the convolution parameters. If many “tunes” => more extrema..
  - Keep only two of them.. Use “knowledge” to select them if more than two.
- Extremum search is in the top level loop.. => loop once over all channels. Integrate, differentiate, select -> done.

A Fast Algorithm: Implementation & Performance.

- *In C++*
- *Can be optimized..*
- *On a Sun "not from too distant past"*
- *From 400 channels, takes 1 mSec to fit the previously shown "Uncoalesced" spectrum …*
- *Same speed on Coalesced, but less precision..*

A Fast Algorithm: Code - I

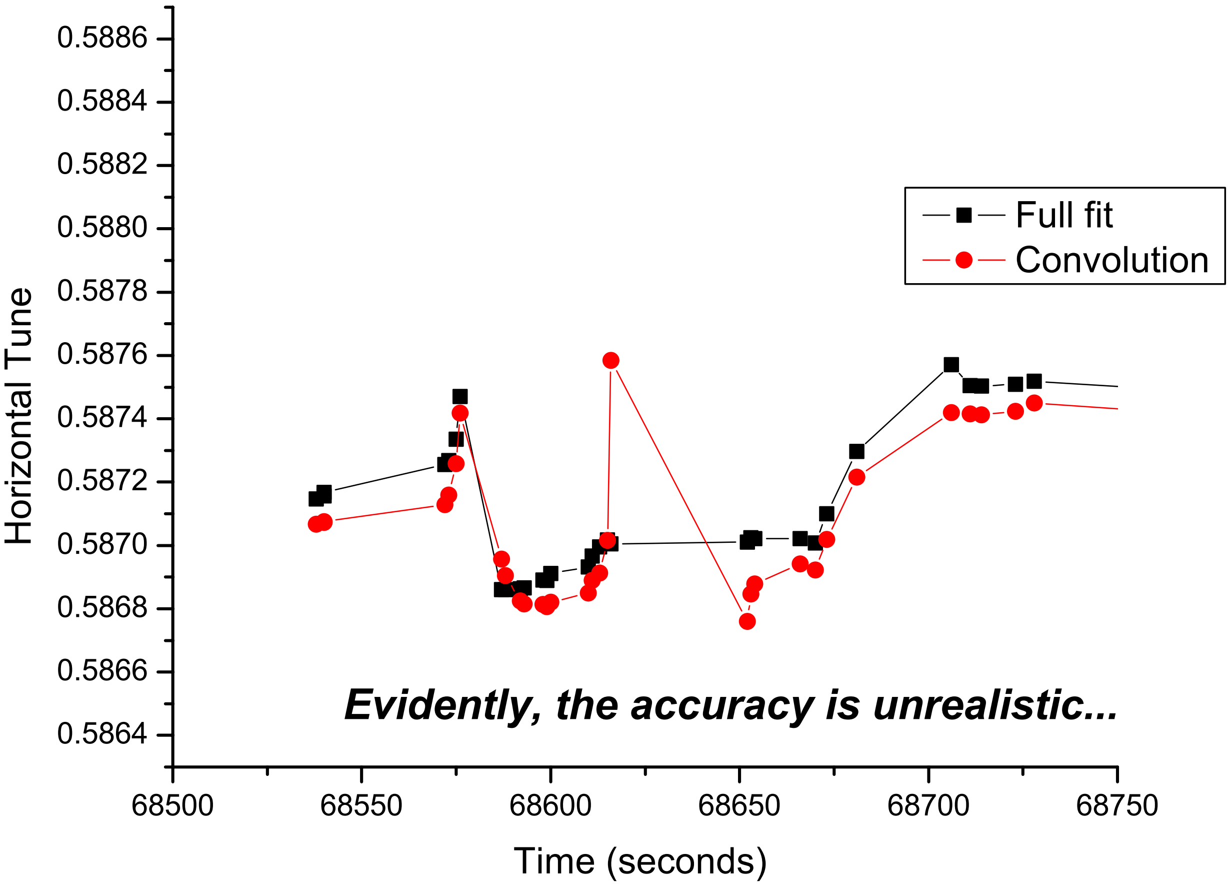
```
for (int i=((int) nBinSmooth)/2; i< (len - ((int) nBinSmooth) + 1); i++) {
    int iStart = i - i3;
    int iStartW = 0;
    if (iStart < 0) {
       iStartW = -iStart;
       iStart = 0;
    }
    int iEnd = i + i3;
    if (iEnd > len) iEnd = len; // Should not be needed
    double val = 0.;
    int iW=iStartW;
    for (int k=iStart; k<iEnd; k++) {
      val += dataIn[k]*weights[iW];
      iW++;
    }
    double deriv = val - prevVal;
    // Now we look for an extremum, if we are at leas one sigma convol
    // away from start
    if (i < nBinSmooth) {
       prevVal = val;
       prevDeriv = deriv;
       continue;
    }

```

A Fast Algorithm: Code , II

```
if ((val > minValForTune) && ((deriv*prevDeriv) < 0.)) {
         // Refine the tune, by fitting to a parabola.
          // Use the fact that we equal bin spacing, so that the
          // quadratic equation can be linearized
          double y3 = val;
          double y2 = prevVal;
          double y1 = y2 - prevDeriv;
         double dx0 = binWidth * 0.5*(y3-y1)/(2.0 * y2 - y3 -y1);
         double tune = tuneBin0 + (i-1)*binWidth + dx0;
         if (debugIsOn) {
              *rollingLog << " Tentative tune at  " << tune << endl;
              *rollingLog << " y1 " << y1 << " y2  " << y2 << "  y3  " << y3 << endl;
            }
          if (abs(tune-tunePrevSet) >  minTuneSep) {
            if (debugIsOn) *rollingLog << " Valid extremum  " << endl;
              extrFound = true;
              if (tuneLowSet && (abs(tune-tunePrevHigh) < maxTuneJump) ) {
                 tuneHigh = tune;
                 amplHigh = val;
               if (debugIsOn) *rollingLog << " Tune High set at  " << tune << endl;
              } else if (abs(tune-tunePrevLow) < maxTuneJump){
               if (debugIsOn) *rollingLog << " Tune Low set at  " << tune << endl;
                 tuneLowSet = true;
                 tuneLow = tune;
                 amplLow = val;
              } else if (abs(tune-tunePrevHigh) < maxTuneJump){
               if (debugIsOn) *rollingLog << " Tune High set at  " << tune << endl;
                 tuneHigh = tune;
                 amplHigh = val;
              }
           }
      }
      prevVal = val;
      prevDeriv = deriv;
   }
```

Preliminary test on 1 Hz data from HP3561a..



Evidently, the accuracy is unrealistic...

This study must be repeated with the correct data, from a digitizer/FFT system..

M. Huening is building such a system.

Digital Solutions..

- Numerical Gaussian Convolution and differentiation could be done with analog hardware .. Mixing.. Filtering! High-band pass filter… Or low-band.. Who cares…
- Because digital are intrinsically more maintainable, tunable and robust than advanced analog solution.. (*Bill Foster, May 7, Run-II Commisioning meeting…)*
  - Case in point: Easy to clone this system running at a different rate (10 Hz instead of 20), from the same signal!). And this can be done in //.. No "cross-talk" between individual componenent.
- Evidently, we need to think in both time-domain ("Real time computing" and "frequency domain" (FFT over finite range of frequencies..).

Status

- The code runs on data generated by the Hp3561a..
- Need to try this on data from the fast digital ADC/DSP + FFT spectrum analyzer, at ~ 20 Hz.
- Our first priority, though, is to integrate the existing software to the TeV control system, so that we can use the tune fitter to automate Chromaticity and coupling measurement.